ATARI®
XL ADDENDUM

ATARI HOME COMPUTER SYSTEM

OPERATING SYSTEM MANUAL

Supplement to ATARI 400/800™ Technical Reference Notes

ATARI®
Printed in U.S.A.

TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 APPLICABLE DOCUMENTS

3.0 HOW THE 1200XL COMPARES TO THE A400/800

3.1 The Help Key
3.2 What the Function Keys Do

Cursor Left
Cursor Right
Cursor Up
Cursor Down
Home Cursor
Cursor to Lower Left Corner
Cursor to Beginning of Physical Line
Cursor to End of Physical Line
Keyboard Enable/Disable
Screen DMA Enable/Disable
Key-Click Enable/Disable
Domestic/International Char. Set Select

3.3 Key Redefinition

Contents of the Key Redefinition Table
Reassignment of the function keys only
Non-reassignable Keys and combinations

3.4 User-Alterable Keyboard Auto-Repeat Rate
3.5 Caps/Lowr Key Toggle Action
3.6 LED Initialization
3.7 Power-On Self-Test
3.8 Option Jumpers
3.9 Additional Hardware Screen Modes
3.10 Text Screen Fine Scrolling
3.11 Disk Communications Enhancements
3.12 Power-On Display Enhancement
3.13 Deleted Features

4.0 MEMORY MAP OF THE 1200XL

5.0 ENHANCEMENTS TO THE A400/800 REV. B OPERATING SYSTEM INCORPORATED IN THE 1200XL

Peripheral Handler Additions
General Improvements

6.0 OTHER CHANGES/GENERAL INFORMATION

Improved Handling of OS Database Variables
NTSC/PAL Timing Provisions
1200XL OS ROM Identification and Checksum

APPENDIX A — An Example of Keyboard Reassignment

APPENDIX B — Suggestions for the Construction of a New Character Set for the New Graphics Modes 12-13, and details of memory use and data interpretation for modes 12-15.

APPENDIX C — OS DataBase Changes from REV. B to 1200

1.0 INTRODUCTION

This manual is designed to serve as a supplement to the ATARI 400™ and ATARI 800™ OPERATING SYSTEM MANUAL.

The 1200XL, as shown in sections 3-5, is a technical upgrade of the A800. The operating system for the 1200XL has been written to maintain, as much as possible, compatibility with application programs which have already been developed for the A400/800.

Since the basic hardware which controls the user interface and the display is, for the most part, compatible with the earlier designs, the operating system, except for the enhancements or changes described here, has remained largely the same. Therefore the data contained in the OS manual for the A400/800 is still valid.

This manual has been written to provide the user with data regarding usage of the added features of the 1200XL operating system, with some details about the characteristics of the peripheral devices with which it will operate. Programmers or peripheral developers who require a greater level of detail regarding the handling of peripheral devices should refer to the documents referenced in item 2 of section 2 below.

2.0 APPLICABLE DOCUMENTS

1. ATARI Home Computer Operating Systems Manual.

   Describes the OS for the A400 and A800, which is the basis for the enhancements described in this manual.

2. ATARI Home Computer Hardware Manual and 1200XL Supplement.

   The Hardware Manual covers the hardware registers which control the various functions of the A400 and A800. The supplement to the hardware manual covers the added features for control of the 1200XL Home Computer. Details that are appropriate to the OS handling of such hardware registers are contained in this OS manual. The user who has need for other hardware-related data should refer to the hardware manual for more information.

3. DE RE ATARI

   This document provides the user with an introduction to the effective use of the ATARI Home Computer hardware. Although written to cover the A400/800, the data contained therein is valid for the 1200XL as well.

3.0 HOW THE 1200XL COMPARES TO THE A400/800

The following is a list of the features and functions which will be discussed in this chapter. Each will be explained in a separate section.

In this chapter, you will learn about:

1. The HELP Key
2. The Function Keys
3. How key codes are redefined and which ones cannot be redefined
4. How to alter the key repeat rate
5. The action of the Caps/Lowr Key
6. How the OS initializes the LED's on the keyboard
7. What happens when a cartridge is installed or removed
8. What happens during power-on self-test
9. What the option jumper assignments mean
10. What new screen modes the 1200XL can use
11. How to enable fine scrolling of the text screen
12. How the disk handler has been changed for improved operation
13. What kind of display is now produced at power-up
14. What features have been deleted as compared to the A400 or A800

Each of the items enumerated above corresponds to the paragraph number in this section which follows. For example, item 1 above is covered in paragraph 3.1, item 2 in paragraph 3.2 and so forth.

3.1 The HELP Key

The operating system, while watching the keyboard, will recognize the pressing of the HELP key as a request to set a flag in the OS database. This flag can be read by whichever application program is in control at the time and react accordingly.

The OS treats the help flag in the same way as the BREAK key in that no ATASCII code is produced but a database variable is set. Therefore, if your program is expecting the HELP key to be pressed, you must not only read the keyboard FIFO (hex location 02FC) for incoming ATASCII codes other than Help, but also occasionally check ("poll") the contents of the HELPFG (help flag) database variable to see if Help was requested.

After reading the database location, and deciding what to do, you must "clear" it for the next time the key will be pressed. The OS does not clear it for you. The Help Flag is cleared by storing a zero in its database variable.

The location of this variable is $02DC. The conditions to which it responds are listed below, along with the codes which will be stored in HELPFG.

Hex value	Condition represented
00	The Help flag is cleared. This flag is cleared at initial power-up reset and subsequently, if set, must be cleared by the application program.
11	HELP key alone was pressed.
51	SHIFT-HELP key combination was pressed.
91	CTRL-HELP key combination was pressed.

The HELP key can be used during the power-on display and during the self test feature. See those sections for more information.

3.2 What The FUNCTION Keys Do

NOTE: This section only applies to XL computers with function keys.

The 1200XL is provided with a set of four function keys. You may redefine the ATASCII values which these keys produce if you desire. As a matter of fact the entire keyboard ATASCII output may be redefined as will be seen later. This section shows the normal definition of the F1-F4 keys, their functions and the ATASCII codes which they produce (if any) as a result of the power-on reset assignment. All values in the table below are given in hexadecimal.

FUNCTION KEY ASSIGNMENT SUMMARY

Key	If pressed alone
F1	Produces the Cursor-up function, returns ATASCII 1C
F2	Produces the Cursor-down function, returns ATASCII 1D
F3	Produces the Cursor-left function, returns ATASCII 1E
F4	Produces the Cursor-right function, returns ATASCII 1F

Key	If pressed with SHIFT
F1	See HOME CURSOR below
F2	See CURSOR TO LOWER LEFT CORNER below
F3	See CURSOR TO BEGINNING OF PHYSICAL LINE below
F4	See CURSOR TO FAR RIGHT OF PHYSICAL LINE below

Key	If pressed with CTRL
F1	See KEYBOARD ENABLE/DISABLE below
F2	See SCREEN DMA ENABLE/DISABLE below
F3	See KEY-CLICK ENABLE/DISABLE below
F4	See DOMESTIC/INTERNATIONAL CHARACTER SET below

Key	If pressed with CTRL and SHIFT
F1	Ignored
F2	Ignored
F3	Ignored
F4	Ignored

HOME CURSOR FUNCTION

SHIFT-F1 causes the cursor to move to the home position of the screen as well as producing the default ATASCII code 1C. The default function is reassignable.

CURSOR TO LOWER LEFT CORNER

SHIFT-F2 causes the cursor to move to the lower left corner of the screen as well as producing the default ATASCII code 1D. The default function is reassignable.

CURSOR TO BEGINNING OF PHYSICAL LINE

SHIFT-F3 causes the cursor to move to the far left of the physical line on which it is located (note, not the logical line which, in the screen editor, could be as many as 3 physical lines). This function is performed by the screen editor as well as generating the default ATASCII code 1E. The default function is reassignable.

CURSOR TO FAR RIGHT WITHIN PHYSICAL LINE

SHIFT-F4 causes the cursor to move to the far right side of the physical line on which it is located. This function is performed by the screen editor as well as generating the default ATASCII code 1F. The default function is reassignable.

KEYBOARD ENABLE/DISABLE

CTRL-F1 controls the keyboard enable/disable function. It produces no ATASCII code. This key combination affects the operating system handling of the keyboard and is not reassignable.

CTRL-F1 disables and re-enables all keyboard functions except for the following:

RESET — is the 6502 RESET key, and cannot be disabled

OPTION
START
SELECT — keys are not controlled by the operating system

Each time you press CTRL-F1, the operating system changes the enabled/disabled status to the opposite of what it was when you pressed this combination. In other words, if the OS had disabled the keyboard, LED 1 would be on. If, at that time, you press CTRL-F1, the OS would re-enable the keyboard and turn LED 1 off. The second press of this combination would reverse the process, disabling the keyboard again.

You may monitor or control the keyboard enable or disable function under software control by reading or writing the OS database variable called KEYDIS (hex location 026D). A value of 0 in this location means the keyboard is enabled, and a value of hex FF here means the keyboard is disabled.

SCREEN DMA ENABLE/DISABLE

CTRL-F2 controls the Screen Enable/Disable Direct Memory Access (DMA). It produces no ATASCII code. This key combination affects the operating system handling of the display function. This key combination is not reassignable.

The 1200XL, on power-up, always enables the screen DMA. What this means is that the system will always initialize itself to display anything which has been defined for the screen display during power up. This same screen DMA enable will also occur if you touch any keyboard key other than the CTRL-F2 combination.

Various types of programs which you write may be heavily involved in arithmetic computations. To speed up the processing in the A400 or A800, you may disable the screen DMA. When it is disabled, the ANTIC processor does not steal memory cycles from the 6502 to get its data for the screen. Therefore during disable mode, the screen remains blank. When it is enabled, the full display which you have defined is visible; however, the processor is slowed down by anywhere from 10 to 40 percent as explained in the section on ANTIC DMA in the Atari Hardware Manual.

On the 1200XL, to start the higher speed/ no display function, press the CTRL-F2 key combination. The display will go blank. To restore the display again at any time, you can press any other key.

During your arithmetic calculations, you may be in continuous process of updating the memory area where the display data is contained. You can then get a status of the operation in process at any time simply by pressing any key other than CTRL-F2, then again press CTRL-F2 to re-enter the higher speed mode.

Your program, then, on completion of the calculation, could exercise direct program control over the ANTIC DMA variable to restore the display when the arithmetic intensive part is over.

The DMA control database variable SMDCTL contains status bits for display list memory access as well as player missile data access. When the combination CTRL-F2 is pressed, the OS will save this value, (if it is not already zero) in database variable location DMASAV$02DD). Then the variable SMDCTL will be set to zero. When the combination is pressed again, the original value is restored to SMDCTL from DMASAV, thereby restoring the display. Your program could perform the same process.

KEY-CLICK ENABLE/DISABLE

CTRL-F3 controls the Key-Click enable/disable function. If pressed once, it disables the audible feedback on keystrokes. Pressed again reenables it. This function only affects an OS database variable and produces no ATASCII code. It is not reassignable.

You may control the key click enable/disable from your program. All that needs to be done is to change the same flag which the operating system uses to indicate whether a key click is required. This flag is called NOCLIK. It is one of the OS database variables, contained at location $02DB.

On power up and reset, the operating system initializes this variable to a value of 00, meaning that key click is enabled. This location, when it contains the value $FF, indicates that no key click is desired. The key combination CTRL-F3 toggles it between the values 00 and FF.

DOMESTIC/INTERNATIONAL CHARACTER SELECTION

CTRL-F4 controls the domestic/international character selection. Default is domestic. It affects an OS database variable only and produces no ATASCII code. It is not reassignable. It toggles the display of character sets, changing between the two each time the key combination is pressed. When the international character set is selected, LED number 2 will be lit.

The international version of the character set is located in the ROM beginning at location $CC00 You can cause the international character set to be selected by storing the constant $CC to location $02F4. This is the location CHBAS. The normal character set is located in the ROM starting at $E000. If a program stores $E0 to CHBAS, it selects the display of the normal characters.

If you have defined your own character set, however, pressing CTRL-F4 will display the international character set. This is because the operating system will test CHBAS and find that the value $CC is not there. Therefore $CC must be the next value which is to be used (selects int'l. set). When it tests CHBAS and finds $CC stored there, it knows that $E0 is the next value to use during the toggle between character sets.

Two variables are used to control the character set selection: CHBAS (02F4) and CHSALT (026B). The Screen Editor (E:) and the Display Handler (S:) initialize variables CHBAS and CHSALT at every OPEN command which you issue to either one. CHBAS is initialized to a value of hex E0 and CHSALT is initialized to a value of hex CC.

When you press CTRL-F4, the operating system swaps the values of CHBAS and CHSALT using the OS variable TEMP as the temporary holding point. Once it completes the swap, if CHBAS is equal to CC, it will light LED 2, indicating that the international character set is selected.

3.3 KEY REDEFINITION

You may redefine most of the 1200XL console keys if desired. The redefinition process consists of setting up a pair of tables which can be referenced by the operating system when it translates your keystroke into an ATASCII value.

The two tables are the KEY Definition Table and the Function Key Definition Table. The operating system has a pair of data tables from which the normal definitions are made. You may define your own set of tables however, then simply tell the operating systen where they are located in memory.

One such use of key redefinition might be to experiment with other, possibly more efficient keyboard layouts, such as perhaps the Dvorak keyboard. An example is given in Appendix A of a keyboard redefinition to allow you to do such an experiment. (Over the years, the QWERTY key layout has been the accepted standard, though many people have found DVORAK to be more efficient. This would allow you to try it for yourself.)

CONTENTS OF THE KEY DEFINITION TABLE

This table allows most of the keys of the 1200XL to generate any desired ATASCII code or special internal function. The exceptions to this are listed at the end of this section. To redefine the keys, it is necessary first to define an area in memory where a 192 byte table may be stored.

Into this table, you will store the definitions of the keys which you desire. Later you will tell the operating system where this table is located so that future references may be made to it instead of the standard definition table.

The organization of this table is as follows:

Lower case convert. Group of 64 bytes
Shift plus key Group of 64 bytes
CTRL plus key Group of 64 bytes

KEYTABLE _ START (Starts at user defined address) Table of lower case conversions

Table of uppercase conversions

Table of control key conversions

KEYTABLE _ START + 191.

The reason that each of the subdivisions of the table has 64 bytes in it is that the hardware can generate a total of 64 hardware keycodes. These codes, numbered 00-63 decimal (00-3F hexadecimal) are used to index directly into one of the three keycode tables. Which table is referenced depends on whether the CTRL or SHIFT keys is pressed.

Note that there is no table for the combination of both CTRL and SHIFT. This combination is invalid and is ignored by the operating system.

Each of the three 64 byte subsections of the table has the form:

00 code	Byte 0 contains conversion for key code 00 for key alone, key with CTRL, or key plus SHIFT. Depends on which table is accessed per which keys pressed.
01 code	Byte 1 contains conversion for key code 01
=	=
3F code	Byte 3F contains conversion for key code 3F

The codes which you place in your table will either generate an ATASCII code (for direct character translation) or they will tell the system to perform a specific function. Specifically any code in the range of 80 to 91 hexadecimal will be treated as special by the system. This is illustrated in the table below.

CODES AND THEIR EFFECT ON THE SYSTEM AFTER TRANSLATION

CODE	EFFECT (if any)
00 thru 7F	Used as the ATASCII code only.
92 thru FF	Used as the ATASCII code only.
80	Ignore, invalid key combination.
81	Invert the video output to the screen.
82	Alpha lock/Lower case toggle.
83	Alpha lock
84	Control Lock
85	End of file
86	ATASCII code
87	ATASCII code
88	ATASCII code
89	Key click on/off
8A	Function 1 *
8B	Function 2 *
8C	Function 3 *
8D	Function 4 *

* NOTE: When it sees these keycode translations, it is told to DO the function which is described in the Function Key descriptions. The ATASCII coded generation for the normal and shifted function keys is handled in a different table, whose description follows that for the keycode hardware translate table.

8E	Cursor to home
8F	Cursor to bottom
90	Cursor to the left margin
91	Cursor to the right margin

The table below shows the key cap corresponding to each key code. The physical position of each key switch within the table determines the hardware code which it will generate. To determine what code it is, take the row address of the cap, and add it to the column address. The result is the hexadecimal value returned to the operating system (range 00-3F) for use in the table lookup for that key.

KEYCODE DEFINITIONS TABLE

	0	1	2	3	4	5	6	7
00	L	J	;	F1	F2	K	+	*
08	O		P	U	RET	I	-	=
10	V	HLP	C	F3	F4	B	X	Z
18	4		3	6	ESC	5	2	1
20	,	SPACE	.	N		M	/	)| (
28	R		E	Y	TAB	T	W	Q
30	9		O	7	BACKS	8		
38	F	H	D		CAPS	G	S	A

As an example the key cap "C" is in the table in row 10, column 2. This means that the hardware generates a hardware code 10 + 2 or 12 hexadecimal. Therefore, in the translation tables shown above, the function code or ATASCII code for this character will be stored in the key definition table position $12 for each of the three types of "C" which are valid (c alone, Shifted C, or Control C). You may cause each of these to perform a separate function or generate a separate ATASCII code by revising the tables.

When you have decided on how you want your keys to be redefined , you tell the operating system where it may find the definitions by storing the address of those definitions in locations 79 and 7A hexadecimal. The low byte of the hexadecimal address where you have stored the keys should be placed in location 79, the high byte is location 7A. This is defined as one of the system vectors, called KEYDEF. It will point to the default, or original key definition table at power-on reset time.

REASSIGNMENT OF THE FUNCTION KEYS ONLY

There may be times when you only want to redefine the function keys and not redefine the rest of the keyboard. The 1200XL operating system allows you to redefine only the function keys by setting up an 8-byte table in place of the 192 byte table which would have otherwise been required. The format of this table is as follows:

F1	← Lowest memory location of the table
F2	
F3	
F4	
SHIFT-F1	
SHIFT-F2	
SHIFT-F3	
SHIFT-F4	← Highest memory location of the table

When you have decided what functions each combination must perform and have built the table, change the system vector FKDEF to point to the lowest address of your table. This vector is located at memory locations 60 and 61 hexadecimal. Location 60 gets the low byte of the hex address, location 61 gets the high byte.

The same codes described in the section titled "CODES AND THEIR EFFECT ON THE SYSTEM AFTER TRANSLATION" are used in this table. However, DO NOT assign codes 8A through 8D to the same function as the key itself. In other words, do not specify that the key F1 should perform function F1, etc. since this would result in an infinite loop. (F1 sensed by the OS sends it to the function key table, which tells it to look up and perform the F1 function, which sends it to the table, and so on, with no possible exit.)

NON-REASSIGNABLE KEYS AND KEY COMBINATIONS

The following keys or key combinations are either specifically wired for special functions or are subjected to special handling by the operating system.

Even though there might be a hardware-generated key code shown in the table above, and a corresponding space in the translate tables, there is no way to reassign these functions. This is because the operating system traps the hardware code directly to perform the specified function and it never gets to the translate mode. These keys or combinations are as follows:

BREAK — This function is fixed as a special case in the operating system. It is sensed by the hardware.

SHIFT — This key is an integral part of the hardware encoding of any key function.

CTRL — This key in an integral part of the hardware encoding of any key function.

OPTION, SELECT, START — All of these are directly wired to and are sensed by the GTIA circuitry.

RESET — Directly wired to the 6502 reset line.

HELP — Function is fixed by the operating system. The help function handling is described elsewhere in this manual.

CTRL-1 — Controls the screen output start/stop function.

CTRL-F1 — See KEYBOARD ENABLE/DISABLE above. As noted there, this function is not reassignable.

CTRL-F2 — See SCREEN DMA CONTROL above. As noted there, this function is not reassignable.

CTRL-F3 — See KEY-CLICK ENABLE/DISABLE above. As noted there, this function is not reassignable.

CTRL-F4 — See DOMESTIC/INTERNATIONAL CHARACTER SET above.

3.4 USER-ALTERABLE KEY AUTO-REPEAT RATE

The 1200XL operating system allows you to control the rate at which a key, continuously held down, will repeat its entry to the system. This change can be done by modifying the OS database variable KEYREP, located at hex address 02DA.

This variable determines the repetition rate by counting the number of VBLANK (vertical blanking) intervals which occur. For the NTSC (60 Hz) system, the initial value of this variable is 6; for PAL systems, the value is 5. This assures a uniform repeat rate of 10 characters per second for either system. The key repeat rate equals the VBLANK rate (60 or 50 per second) divided by the KEYREP value.

Under control of this variable, the maximum "controllable" key repeat rate would be 50 characters per second on the PAL, and 60 characters per second on the NTSC (screen refresh rate). This would occur with a value of 1 in this variable.

You may control the rate at which occurs before the key repeat starts. The OS database variable which controls this is called KRPDEL. Its hex address is 02D9.

It controls the number of VBLANKs which must occur between the sensing of the key pressed until the first repeat occurs. From that time on, the repeat rate is controlled as described above. The initial values used by the OS provide a 0.8 second initial delay for either NTSC (count = 48) or PAL (count = 40) systems.

3.5 CAPS/LOWR KEY TOGGLE ACTION

The CAPS/LOWR key on the 1200XL functions as shown in the chart below:

KEY COMBINATION	CURRENT STATE	NEW STATE
CAPS	Control Lock	Lower Case
CAPS	Alpha Lock	Lower Case
CAPS	Lower Case	Alpha Lock
SHIFT-CAPS	— any —	Alpha Lock
CTRL-CAPS	— any —	Control Lock
CTRL-SHIFT-CAPS	— any —	— no change —

The meaning of the terms is as follows:

Lower Case — All key caps respond in lower case mode

Alpha Lock — All alphabetic keys (A-Z) respond in upper case mode, all others lower case

Control Lock — All alphabetic keys (A-Z) respond as though the control key is being held down as well as the selected key

3.6 LED INITIALIZATION

The 1200XL has two LED's on the front panel, called LED 1, and LED 2. LED 1, when lit, indicates that the Keyboard is disabled. LED 2, when lit, indicates that the international character set is selected. The operating system enables the keyboard and selects the domestic character set on power up and reset. Therefore these LED's will both be off.

3.7 POWER-ON SELF-TEST

During the initial power-on, the 1200XL operating system will perform the following quick check of the integrity of the system RAM and ROM:

a. Is it possible to write $FF (all ones) to all RAM locations?

b. Is it possible to write $00 (all zeros) to all RAM locations?

c. Does a checksum of the two ROM's compare to that stored within each ROM?

If any of these tests fail, the operating system will transfer control to the self-test memory test routine. Here a more thorough test of both RAM and ROM can take place.

3.8 OPTION JUMPERS

The 1200XL is provided with a set of four hardware jumpers which are designed to tell the operating system how the system is configured. As of the date of this writing, only one of the four jumpers has been assigned, specifically J1. This is specified in the table below. During the power-on sequence, the 1200XL operating system reads the state of these jumpers and stores this state in the OS database variable JMPERS, location 03OE.

The bit assignments for each of the four jumpers is as specified below. The bits are all active low, meaning that if a line reads a digital zero, the jumper is installed.

BIT	FUNCTION	HARDWARE NAME
0	Self test enable (will run self test if low)	J1 (pot 4)
1-3	Reserved for future use	
4-7	Unused	

3.9 ADDITIONAL HARDWARE SCREEN MODES

The 1200XL adds direct access to the remaining special purpose display processor operating modes. The table below shows the current mapping which has been provided for the A400 and A800. The table which follows thereafter shows the added modes and the numbers which the software can use to access the extra modes.

Mode mapping common to A400/A800:

Software Mode		ANTIC MODE		GTIA MODE
0	($00)	2	($02)	0
1	($01)	6	($06)	0
2	($02)	7	($07)	0
3	($03)	8	($08)	0
4	($04)	9	($09)	0
5	($05)	10	($0A)	0
6	($06)	11	($0B)	0
7	($07)	13	($0D)	0
8	($08)	15	($0F)	0
9	($09)	15	($0F)	1
10	($0A)	15	($0F)	2
11	($0B)	15	($0F)	3

Mode mapping for 1200XL (additional):

Software Mode		ANTIC MODE		GTIA MODE
12	($0C)	4	($04)	0 (note 1)
13	($0D)	5	($05)	0 (note 1)
14	($0E)	12	($0C)	0
15	($0F)	14	($0E)	0

Note 1: The existing character sets will not provide recognizable characters for these new modes. Therefore you will have to provide the character set if you use these modes. This is done by defining the full character set, then modifying the OS database variable CHBAS to point to the most significant byte of the address at which the character set starts. CHBAS is located at $2F4.

Appendix B of this manual contains some suggestions on the method for designing a new character set to support those added modes.

3.10 TEXT SCREEN FINE SCROLLING

The screen editor (E:) supports fine scrolling of the text screen data as an option. This fine scrolling option will be enabled if the database variable FINE (hex location 026E) is set nonzero prior to issuing the OPEN command to the screen editor. Likewise, the feature will be disabled if this location is set to 00 before issuing the OPEN.

There are only two allowed values for FINE = 0 and hex FF. Other values may produce undesirable results.

During an OPEN command to the Screen Editor (E:), if FINE (026E) is hex FF, then a fine scrolling display list is created. This display list will be one byte larger than a coarse scrolling display list. In addition, the OS places the address of a display list interrupt routine into the display list vector VDSLST (0200) replacing an other vector which you might have already stored there.

When fine scrolling is enabled, the Screen Editor's display list interrupt service routine modifies the content of color register COLPF1 (D017) for the very last visible line of the screen.

When a CLOSE command is issued for the Screen Editor, if FINE is hex FF, then the address of an RTI is placed into the display list vector VDSLST (0200). For OS versions 11 and beyond, FINE is set to zero again, and the screen is reopened with a coarse scrolling display list.

The recommended manner for enabling and disabling fine scrolling is shown below:

a. Set FINE to hex FF

b. OPEN E: using an IOCB number

c. Use E: as usual; fine scrolling is enabled

d. CLOSE E:

e. If the IOCB is now open, then you are finished, otherwise continue with the next step

f. Set FINE to zero

g. OPEN E:

3.11 DISK COMMUNICATIONS ENHANCEMENTS

The 1200XL adds the capability for the resident disk handler to read and write disk sectors having variable length from 1 to 65536 bytes. The default length, as is used on the A400 and A800 currently, is 128 bytes. Both at power-on and RESET (warm start), the 128 byte sector length is established. Your program can alter this length by modifying the OS database variable DSCTLN. The location of this two-byte variable is 02D5 and 02D6 (lo byte in 02D5, hi in 02D6).

In addition to the capability to read and write variable length sectors, the 1200XL also adds the capability to write a sector to the disk without a read-verify operation always following it. This is the command 'P' which was specifically excluded in the previous releases of the operating system.

With this capability added, you have a choice of either using the verify, for system integrity (always read after write). Or you can take a chance of writing a bad sector on rare occasions but increasing your average speed of disk usage by some value related to the verify time. You may want to experiment with some of your programs with and without verify to see the results.

3.12 POWER-ON DISPLAY ENHANCEMENT

In place of the original power-on memo pad display used by the A400 and A800 (in the absence of a cartridge or disk), the 1200XL displays a dynamic ATARI rainbow. If you press the HELP key while the rainbow is displayed, the 1200XL will enter the self-test mode.

4.0 MEMORY MAP OF THE 1200XL

The following table shows how the 6502 processor perceives the various address spaces which it can access. The maximum allowable address range, with the 16 bit address of the 6502 is hexadecimal 0000-FFFF. This address range is split, by the hardware memory management circuitry, as follows:

(Note: The 1200XL uses 64K RAM's as the main system writeable memory. Addresses within those RAM's, which would normally have filled the entire memory access space of 0000-FFFF of the processor, are prevented from access by the memory manager. This allows ROM's, cartridge memory, and peripherals to occupy a part of the memory space as is noted below.)

1200XL MEMORY MAP

HEX ADDRESS	WHAT IS ACCESSED THERE	NOTES
FFFF-D800	OS-ROM or RAM if ROM disabled	1
D7FF-D000	The special purpose chips respond to the address ranges shown in the listing below	
	D000-D0FF GTIA D200-D2FF POKEY D300-D3FF PIA D400-D4FF ANTIC D500-D5FF Any read or write to an address in this range enables the cartridge control line CCNTL on the cartridge interface (same as A400/A800.	
	D100-D1FF, D600-D6FF, and D700-D7FF are reserved for future use.	
	OS-ROM physically present, but cannot be accessed here.	2
CFFF-C000	OS-ROM or RAM if ROM is disabled	1
BFFF-A000	RAM, or cartridge interface	3
9FFF-8000	RAM, or cartridge interface	3
7FFF-5800	RAM	
57FF-5000	RAM, unless in self-test mode	2
4FFF-0000	RAM	

NOTES: 1. Access to the OS ROM may be disabled by wirting a zero to port B of the PIA, bit O. Access is normally enabled, with a 1 present in this bit. (When changing this bit in the register, other bits should not be changed.)

2. The self-test ROM code is physically present in the OS ROM at actual address DOOO-D7FF. However, this area is used for the access to the memory mapped I/O devices. When the self-test feature is invoked, the RAM located from 5OOO-57FF is disabled. The memory manager re-maps the memory access such that the OS ROM physical addresses DOOO-D7FF are accessed at 5OOO-57FF. The memory manager uses port B of the PIA, bit to determine whether to access RAM or ROM in the region 5OOO-57F. If bit 7 is high, RAM is accessed. If bit 7 is low, the OS-ROM is accessed instead. (When changing this bit in the register, other bits should not be changed.)

(Port B was used in the A4OO/8OO to service the game ports 3 and 4. The use of the remaining bits of ths port are specified in Section 6 of this manual.)

3. ROM will be selected in these regions if control lines RD4 or RD5 are pulled up to +5V by the cartridge. RD4 controls ROM select in the region 8OOO-9FFF. RD5 controls ROM select in the region AOOO-BFFF.

5.0 ENHANCEMENTS TO THE A400/800 REV. B OPERATING SYSTEM INCORPORATED IN THE 1200XL

This section describes a set of enhancements which include new methods of handling peripheral products and, in a separate section, improvements in basic operations of the system.The latter might be referred to as "bug fixes".

PERIPHERAL HANDLER ADDITIONS

To accommodate a new class of peripheral devices, the operating system now includes a relocating loader, used to upload peripheral handlers through the serial (I/O interface).

In the A400/800, device handlers for the peripherals were uploaded as fixed location (absolute) object code. These handlers were loaded using a set of device inquires, or polls, known as types 0, 1 and 2. Information on types 0, 1 and 2 Poll Commands is available from Atari Customer Service.

The 1200XL adds two other types of polls to its operating system. One poll, known as type 3, is issued at power-on or reset time. The other, type 4, can be issued as a result of an OPEN command by an application program.

Type 3 Poll Command

The type 3 poll command itself is used as an "Are You There?" type of command. Associated with the type 3 poll are two other types, specifically the:

a) Poll Reset

and b) Null Poll

Poll Reset consists of the following SIO command byte sequence (refer to the SIO document for further explanation of the byte types):

Byte Position	Value (hex)
Device Address	4F
Command Byte	40
AUX1	4F
AUX2	4F
Command Checksum	Normal (checked by peripheral)

The 4F in AUX1 and AUX2 define this sequence to all peripherals as a poll reset.

After responding to a type 3 poll by sending a handler to the system, a peripheral is not supposed to respond again to a type 3 poll. The Poll Reset command, at power-up, resets all type 3 peripherals, freeing them to respond to the poll request. However, no serial bus device sends back any data as a result of a poll reset command.

Type 3 Poll (Are you there?)

There may be several types of peripherals which can respond to a type 3 poll. In types 0, 1 and 2, the device address sent on the serial line specifies which exact device is being called. In the type 3 poll processing, however, the address remains fixed (4F) and the devices each respond after a specific number of poll 3 retries. In other words, during poll 3 operations, the computer doesn't know which peripherals are actually attached, but will keep asking "is anybody there" until it has reached its last retry and no peripheral has responded.

Each peripheral which does respond to the type 3 poll must be designed to count the number of retries of type 3 polls, then to respond as described below on its own specified retry slot. Each time it sees a command other than a type 3 poll, these peripherals must reset their retry counters. This allows the computer to load the handler for each peripheral which responds, then restart its poll 3 sequence (original retry number restored) to look for another poll 3 response from the next peripheral (if any).

Since each peripheral responds only once (after a poll reset), a second request at a specific retry slot causes no peripheral response and allows the next retry slot to be polled.

This poll ("are you there?") is sent as follows:

Byte Position	Value (hex)
Device address	4F
Command Byte	40
AUX1	00
AUX2	00
Command checksum	Normal (checked by peripheral)

When, after checking the retry count, it is a peripheral's turn to respond, it sends back the following data to the computer on the serial interface:

a) An ACK response byte, and

b) 1. Low byte of handler size in bytes (must be EVEN)
   2. High byte of handler size
   3. Device Serial I/O Address to be used for loading
   4. Peripheral Revision Number

These four bytes, if sent by the peripheral, will be stored in OS variables DVSTAT (02EA hex) through DVSTAT+3. If there is a successful return to the OS (not a timeout or other problem), it indicates that there is a handler to be loaded. The loading is performed, then the type 3 poll is repeated until all retries are exhausted and no peripheral responds.

Once the device address data is received from the peripheral during this type 3 poll, it can thereafter be referenced directly on the serial bus by its address in place of the original poll address 4F.

Specific details of the actions taken by the OS after receiving an answer from a peripheral may be found in Appendix C.

Null Poll Command

This command is used as a serial bus no-operation. If any error should occur during loading of a peripheral handler or by the relocator, the system should be free to "back out" of the linking of the faulty loader and tell the peripherals that it is ready for the next one to be loaded. Since this null poll is a non-type-3 poll, all peripherals will have reset their retry counters and should be ready for another sequence of retries, looking for their own response retry slot. This maintains synchronization between the computer and the peripherals.

The structure of the Null Poll is as follows:

Byte Position	Value (hex)
Device Address	4F
Command Byte	4O
AUX1	4E
AUX2	4E
Command Checksum	Normal (peripherals check it)

Type 4 Polling

This type of poll is sent out on the serial bus as a result of an application initiated request. During an OPEN command, a device which responds to a type 4 poll may conditionally or unconditionally be polled to determine if it is online and may or may not have its handler uploaded and linked to the system under control of the OS. Detailed information regarding the handling of the device under various operating conditions may be found in Appendix C.

The Type 4 Poll is a serial port command structured as follows:

- Device address of 4F hex (peripherals looking for Type 4 Poll may ignore the device address and look only for the poll command '@'; however, the device address will always be 4F hex and the peripheral may check this);
- Command is '@' (4O hex) (peripherals looking for this poll will always look for the '@' command);
- AUX1 contains the device name, which is an ATASCII upper-case letter (range 41 hex through 5A hex) (the peripheral must be assigned that device name in order to legally answer the poll);
- AUX2 contains the device number, which is an ATASCII digit (range ATASCII 1 through 9, 31 hex through 39 hex) (the peripheral may optionally use this information in deciding whethe or not to answer the poll);
- Standard command checksum (peripheral checks this).

This poll differs from the Type 3 Poll in that the device name and number is included in the poll. Therefore the peripheral need not count retries of the type 4 poll and should answer the poll as soon as the poll command is recognized. There is no limitation on the type 4 poll; the peripheral should answer its type 4 poll each time it is issued.

The peripheral response to a type 4 poll is the same as for the type 3 poll. The four response bytes are placed, by the computer, into DVSTAT through DVSTAT+3 (O2EA through O2ED hex.).

GENERAL ENHANCEMENTS TO THE REV. B OS FUNCTIONS

The following functions which are supported by the A400/800 Rev. B Operating System have been further enhanced by the addition of the following features:

Printer CLOSE with data in the buffer —

The printer handler will insert an EOL(end-of-line) character in the printer buffer, if one is not there, before sending the buffer to the printer on a CLOSE. This assures that the last line will be printed immediately rather than having the printer forced offline to output the final line.

Printer Unit Number Handling —

The printer handler has been changed so that it will process the unit number in the IOCB, allowing separate addressing for printers P1 through P8.

CIO Handling of Truncated Records on Read —

The CIO now places an EOL in the user's input buffer on the occurrence of either a record longer than the buffer being read or an EOF being encountered during the read attempt. This assures that all records are accessible, even if the user has not provided a sufficient buffer size, he will at least get as much of the record as he has provided for.

CIO Error Handling With Zero Length Buffer —

The CIO will return a buffer length of zero (in the 6502 A-register) when there is a handler error while effecting a zero length buffer transfer. (See CIO section in the OS manual.)

Display Handler Cursor Handling —

The display handler now accepts a screen clear code no matter what value is in the cursor X and Y coordinates.

Display Handler/Screen Editor Memory Clearing —

The Display handler and Screen editor will not clear memory beyond the end of memory as indicated by RAMTOP. Now it is possible for the user to specify the top of memory to be used by the system and to store device handlers or personal machine code in the memory area above the display. Changing display graphics modes, then, will not erase any data which has been placed in the RAM area above that assigned for use by the display or screen editor.

Rework of the Floating Point Package —

The 1200XL operating system corrects a bug in the Rev. B OS. It now produces an error status when an attempt is made to calculate the LOG or LOG10 of zero.

New ROM Vectors —

The following fixed entry point vectors have been added to the 1200XL ROM set:

E480	JMP PUPDIS	entry to power-on display
E483	JMP SLFTST	entry to the self-test pgm.
E486	JMP PHENTR	entry to uploaded handler enter.
E489	JMP PHULNK	entry to uploaded handler unlink.
E48C	JMP PHINIS	entry to uploaded handler init.

6.0 OTHER CHANGES/GENERAL INFORMATION

This section deals with items which involve operating system changes, but which do not easily fit into any other category.

IMPROVED HANDLING OF OS DATABASE VARIABLES

During normal power-on sequence (cold start), the OS database variables from $O3ED-$O3FF are set to zero. During a RESET (warm start), they are NOT changed by the OS. This means that an enhanced version of the operating system in the future will be able to make use of these locations without reloading them after any RESET operation.

These bytes are all reserved for use in future OS revisions.

NTSC/PAL VERSION TIMING PROVISIONS

There are various timing differences between the NTSC (60 hz) and the PAL (50 hz) versions. To eliminate the necessity for providing a special operating system ROM set for each one, the specific timing adjustment values are handled within the single ROM set.

To determine which type of system the ROM is operating on, the operating system checks a flag within the GTIA chip and adjusts all timings accordingly. This was possible because the GTIA must be different to handle the modified display format for the 50 Hz version. By making certain timings a function of the state of this flag, it was possible to make external timings independent of the NTSC or PAL system itself.

The timing values relate to the handling of the 115 Volt cassette player (Atari 410) and the console auto-repeat rate as shown in the table below:

CASSETTE TIMINGS NOW INDEPENDENT	TIMING
Write Inter-record gap (long)	3.0 sec.
Read IRG delay (long)	2.0 sec.
Write IRG (short)	0.25 sec.
Read IRG delay (short)	0.16 sec.
Write File leader	19.2 sec.
Read Leader delay	9.6 sec.
Beep cue duration	0.5 sec.
Beep cue separation	0.16 sec.

AUTO-REPEAT FUNCTIONS NOW INDEPENDENT	TIMING
Initial delay for auto-repeat	0.8 sec.
Repeat rate	10.0 char/sec.

1200XL OS ROM IDENTIFICATION AND CHECKSUM DATA

Each of the two ROM's in which the 1200XL operating system is contained has a capacity of 64K bits organized as 8K by 8. Within each of the ROM's is a block of data organized as shown in the diagram below, to identify the ROM and to give its checksum. The checksum is tested by the operating system as part of the power up sequence.

The format of the block for the COOO-DFFF ROM is as follows:

ROM Cksum (lo)		COOO	Checksum which is the sum of all bytes in ROM except checksum bytes themselves.
ROM Cksum (hi)		COOl	
Dl	Dl	COO2	Revision date having the form DDMMYY where D = day digit M = month digit Y = year digit Each a 4 bit BCD digit. Reserved contains $OO for the 12OOXL.
Ml	M2	COO3	
Yl	Y2	COO4	
Option byte		COO5	
Al		COO6	Part number having the form AANNNNNN, where A's represent ASCII characters, N are BCD digits.
A2		COO7	
Nl	N2	COO8	
N3	N4	COO9	
N5	N6	COOA	
Revision No.		COOB	

The format of the identification block for the EOOO-FFFF ROM is as follows:

Dl	D2	FFEE	Revision date having the form DDMMYY where D = day digit M = month digit Y = year digit Each a 4 bit BCD digit. Hardware product identifier, will be used by Atari to identify Home Computer products, for 12OOXL = $Ol.
Ml	M2	FFEF	
Yl	Y2	FFFO	
Option byte		FFFl	
Al		FFF2	Part number having the form AANNNNNN, where A's represent ASCII characters, N are BCD digits.
A2		FFF3	
N l	N 2	FFF4	
N3	N4	FFF5	
N5	N6	FFF6	
Revision No.		FFF7	
ROM Cksum (lo)		FFF8	Checksum which is the sum of all bytes in ROM except for checksum bytes themselves.
ROM Cksum (hi)		FFF9	
vector table for for NMI, RES and IRQ		FFFA-FFFF	This area reserved for power-on reset vectors, NMI and IRQ vectors.

PORT B CHANGES

Port B of the PIA is a read/write port which no longer is connected to game I/O ports. Instead, its bits control various functions which include control of LED 1, LED 2, read enable of the the OS ROM's and other functions. To change only one single bit at a time within that port, the following technique should be used.

```
Clear A Bit (bit b)
    LDA PORTB
    AND # $FF-b
    STA PORTB ;clears only bit b in the port

Set A Bit (bit b)
    LDA PORTB
    ORA # b
    STA PORTB ;sets only bit b in the port
```

XL PORTB ($D301) BIT ASSIGNMENTS		
BIT	VALUE	USE
0	0	OS ROM DISABLED, RAM ENABLED
	1	OS ROM ENABLED
		The memory region mapped to the OS ROM is from $C000 to $FFFF except for the region from $D000 to $D7FF which is always mapped to the hardware I/O chips (GTIA, POKEY, PIA, ANTIC).
1	0	BASIC ENABLED
	1	BASIC DISABLED, RAM ENABLED
		The memory region mapped to BASIC is from $BFFF.
2	0	LED #1 ON
	1	LED #1 OFF
3	0	LED #2 ON
	1	LED #2 OFF
4		RESERVED FOR FUTURE USE
5		RESERVED FOR FUTURE USE
6		RESERVED FOR FUTURE USE
7	0	SELF TEST ROM ENABLED
	1	SELF TEST ROM DISABLED, RAM ENABLED
		The memory region mapped to the self test ROM is from $5000 to $57FF.

NOTE: The OS VBLANK process copies the port A joystick and paddle values into the Port B shadows. Thus, stick 0 affects both 0 and 2, stick 1 affects both 1 and 3.

REV-LEVEL DETERMINATION

To allow program products to determine which Atari Home Computer and Operating System Revision level it is operating with, the following tests are recommended:

If location $FCD8 = $A2, then product is an A400/A800 wherein:

If location $FFF8 = $DD and $FFF9 = $57
then OS is NTSC rev A.
If location $FFF8 = $D6 and $FFF9 = $57
then OS is PAL rev A.
If location $FFF8 = $F3 and $FFF9 = $E6
then OS is NTSC rev B.
If location $FFF8 = $22 and $FFF9 = $58
then OS is PAL rev B.
Otherwise, it is some future A400/A800 OS.

If location $FCD8 not $A2, then product is a 1200XL or other future home computer product, wherein:

If location $FFF1 = $01, then OS is 1200XL
and location $FFF7 will be the
internal rev number for the 1200XL OS.

Otherwise, location $FFF1 = product code for
future Atari Home Computer product
and location $FFF7 contains OS rev
level for this product.

APPENDIX A — AN EXAMPLE OF KEYBOARD REASSIGNMENT

As suggested earlier in this document, the keyboard functions may be reassigned. The table below gives the corresponding keys for the Dvorak (also known as the American Simplified) Keyboard. When the typewriter was first invented in 1867, Christopher L. Sholes chose a layout for the keys which would slow down the good typists of his day and thereby prevent his machine from jamming. This keyboard has endured to this day.

In 1932, August Dvorak invented this key layout which places the most often used characters, including the vowels, on the "home" key line and also redistributes the keystrokes from a 60-70% left-hand activity to an almost 50/50 activity. Certain manufacturers currently offer this key layout as an option. Now you can try it for yourself if you wish. Only the list of key correspondence is given here. It is left to the reader to compose the key function table using the data contained earlier in this manual.

TOP ROW OF KEYBOARD		CENTER ROW		BOTTOM ROW	
Current	Dvorak	Current	Dvorak	Current	Dvorak
Q	?	A	A	Z	;
q	/			z	;
W	,	S	O	X	Q
w	,				
E	.	D	E	C	J
e	.				
R	P	F	U	V	K
T	Y	G	I	B	X
Y	F	H	D	N	B
U	G	J	H	M	M
I	C	K	T	"	W
				"	w
O	R	L	N	.	V
				.	v
P	L	:	S	?	Z
		;	s	/	z
¼	"	"	_(underline)		
½	'	'	—		

APPENDIX B — SUGGESTIONS FOR THE CONSTRUCTION OF A NEW CHARACTER SET FOR THE NEW GRAPHICS MODES

This appendix covers the new graphics modes 12, 13, 14 and 15 now provided on the 1200XL. Modes 14 and 15 are pure graphics modes with resolutions of 160 by 20 and 160 by 40 respectively. Since these are not character modes, the discussion below will be limited only to modes 12 and 13.

Graphics 12 and 13 do not produce recognizable characters, for the most part, using the standard character set. One will understand why this is true by examining the following comparison between Graphics mode 0 to 12 and 13.

> Mode 0 is a 40 character mode. Each character is formed from an 8 wide by 8 high pixel matrix. Each pixel is one bit wide in memory and is ½ of a color clock wide on the screen.
>
> Modes 12 and 13 are also 40 character modes. However, each character is formed from a 4 wide by 8 high pixel matrix, with each pixel 2 bits wide in memory and one color clock wide on the screen. This forces the character to be the same width as that used in Graphics mode 0, but cannot convey the same information within 4 pixels as with 8 as far as character recognition is concerned. (It is difficult to form a recognizable character in a four by eight dot matrix).

Let's examine how the 4-pixel character is formed, again comparing the way the 8-pixel character is formed in mode 0.

> Mode 0 has a choice of two colors for each pixel (the hardware manual says 1 ½ colors, but it is actually either the hue and luminance of playfield 2 if there is a zero bit in the selected pixel position, or the hue from playfield 2 with the luminance of playfield 1 if there is a 1 bit in the selected pixel position. Therefore, each single bit in the character definition byte for a given line occupies a single ½-color-clock-wide pixel position. The character set built into the OS defines the characters in an 8 by 8 matrix.
>
> Mode 12 also uses 8 scan lines per character. However, it uses the character bytes in a different manner. Each of the character bytes retrieved by the ANTIC is treated as a set of four two-bit quantities, where each bit pair describes the color which is to be applied to one of the 4 single color-clock-wide pixels which are part of the character. Mode 13 is the same in its treatment of the data bytes, but each of the characters is double-height (16 scan lines instead of 8) and each data byte is used twice which effectively doubles the height of the character.

Let's look at a typical character, for example a W. The bits which form a W in the default character set are similar to the following:

1	0	0	0	0	0	0	1	display:
1	0	0	0	0	0	0	1	
1	0	0	1	1	0	0	1	
1	0	0	1	1	0	0	1	
1	0	1	0	0	1	0	1	
1	1	0	0	0	0	1	1	
1	1	0	0	0	0	1	1	
1	0	0	0	0	0	0	1	

(**NOTE:** This is not the exact representation, but is used as an example of correct interpretation in mode 0 and incorrect interpretation in modes 12 and 13.)

If you view the sample set of bytes, each at consecutive addresses within the defined character set, it actually looks like a W when you trace the outline formed by the 1's in the byte set, as shown in the display example to the right of the byte representation.

In this mode 0 display, each of the 1's would be one color, and each of the zeros would be another color, assuring a readable display.

For the modes 12 and 13, the four (not 8) pixels are controlled as follows:

If two-bit value is:	Then the pixel color is:
00	the background color
01	the playfield 0 color
10	the playfield 1 color
11	the playfield 2 color (if bit 7 of char = 0)
11	the playfield 3 color (if bit 7 of char = 1)

For the example shown, then, the 4th line from the bottom would display a 10 10 01 01 or 4 pixels of playfield colors 1, 1, 0, 0 in a row, if the standard character set is used. And the bottom-most line would display playfield colors 1, BAK, BAK, 0 in a row. As may be imagined, it is difficult to recognize such a character. (This character is a mirror image left to right — nonsymmetric characters would be even more difficult to recognize.)

To build a character set for these modes 12 and 13, then, it is suggested that you build each character as double wide, to allow a total of 8 pixels (by 8 lines) to define the character. This would also mean assigning two character set locations for each character and treating each character printed in these modes as two characters to be printed. For the example of the W, the character set might look like this:

Byte set 1:				Byte set 2:			
10	00	00	00	00	00	00	10
10	00	00	00	00	00	00	10
10	00	00	10	10	00	00	10
10	00	00	10	10	00	00	10
10	00	10	00	00	10	00	10
10	10	00	00	00	00	10	10
10	10	00	00	00	00	10	10
10	00	00	00	00	00	00	10

Byte set 1 may represent ATASCII value hex 57 within the new character set table, and set 2 may be at ATASCII value hex D7 (hex 57 plus hex 80) if desired. You may feel free, of course, to assign your character sets in any manner you desire.

Therefore, if you would print these two characters side by side on the screen, it would become effectively a 20 character per line mode, with the resultant 10-combination treated as the 1-bit in the mode 0 example and the 00-combination as the 0-bit in the mode 0 example, forming a recognizable W in the process.

Note also that you may want to design these new character sets in a 7 by 7 matrix starting the upper left hand corner of the bit-pair set to allow at least one blank row and column between each of the new characters. (This was not done in the example.)

Thus many combinations of colorful characters may be formed using this technique, allowing the user of the 1200XL additional program flexibility.

MEMORY REQUIREMENTS FOR NEW SCREEN MODES

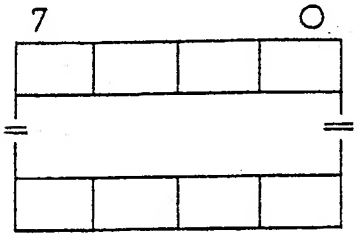
The following table summarizes the memory utilization for the new modes 12 through 15.

Mode No.	Horiz. Posit.	Vert. w/o split screen	Vert. with split screen	Colors	Data Value range	Color Reg. used	Memory Required (split screen)	Memory Required (full screen)
12	40	24	20	5	00-7F	*	1154	1152
13	40	12	10	5	00-7F	*	664	660
14	160	192	160	2	0 1	BAK PF0	4270	4296
15	160	192	160	4	0 1 2 3	BAK PF0 PF1 PF2	8112	8138

*Note: See character definition format for modes 12 and 13.

CHARACTER DEFINITION FORMAT FOR MODES 12 AND 13

The following chart shows the layout for a single character of the character set which would be used for forming characters in modes 12 and 13. As explained above, the value of each of the bit pairs specifies what color will appear on a full-width color clock when this character mode is selected.



Bit positions within this line

Relative byte 0

Relative byte 7

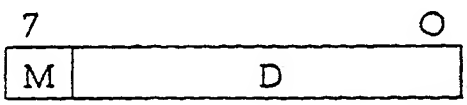
Each 2 bit color specification in the character definition maps to the color registers as follows:

If the bits have the value:	Then the color register used to select the color of the pixel is:
0	BAK
1	PF0
2	PF1
3	PF2 if Bit-7 (the color modifier) equals a 0, or PF3 if Bit-7 (the color modifier) equals a 1.

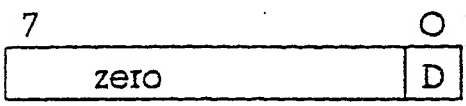
The meaning of the color modifier is shown in the following tables, which show the formats for the data bytes which are used to produce the display itself. As a reminder, the data which is to be displayed on the screen is located somewhere in memory. The data is located by the address provided in the display list. The data bytes themselves in these locations will be interpreted according to the following table.

TABLE of DATA FORMATS used for GET CHARACTER/PUT CHARACTER for MODES 12 through 15.

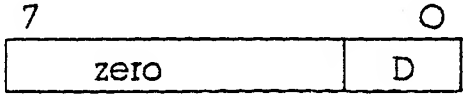
Modes 12, 13 — M = color modifier bit; D = truncated ATASCII



Mode 14 — D = color



Mode 15 — D = color



APPENDIX C — DATA BASE CHANGES FROM REV. B TO 1200

This appendix lists the difference in memory usage between the Rev. B operating system of the A400/800 and the operating system for the 1200XL.

LOCATION	REV. B USE	1200XL USE
0000	reserved	LNFLG — for inhouse debugger
0001	reserved	NGFLAG — for power-up self test
001C	PTIMOT moved (0314)	ABUFPT — reserved
001D	PBPNT moved (02DE)	ABUFPT — reserved
001E	PBUFSZ moved (02DF)	ABUFPT — reserved
001F	PTEMP (deleted)	ABUFPT — reserved
0036	CRETRY moved (029C)	LTEMP — loader temp.
0037	DRETRY moved (02BD)	LTEMP — loader temp.
004A	CKEY moved (03E9)	ZCHAIN — handler loader temp.
004B	CASSBT moved (03E9)	ZCHAIN — handler loader temp.
0060	NEWROW moved (02F5)	FKDEF — func. key def. ptr.
0061	NEWCOL moved (02F6)	FKDEF — func. key def. ptr.
0062	NEWCOL moved (02F7)	PALNTS — PAL/NTSC flag.
0079	ROWINC moved (02F8)	KEYDEF — key def. pointer
007A	COLINC moved (02F9	KEYDEF — key def. pointer
0233	reserved	LCOUNT — loader temp.
0238-0239	reserved	RELADR — loader
0245	reserved	RECLEN — loader
0247	LINBUF (deleted)	reserved
0248-026A	LINBUF (deleted)	reserved
026B	LINBUF (deleted)	CHSALT — character set ptr.
026C	LINBUF (deleted)	VSFLAG — fine scroll temp.
026D	LINBUF (deleted)	KEYDIS — keyboard disable
026E	LINBUF (deleted)	FINE — fine scrolling flag
0288	CSTAT (deleted)	HIBYTE — loader
028E	reserved	NEWADR — loader
029C	TMPX1 (deleted)	CRETRY — from 0036
02BD	HOLD5 (deleted)	DRETRY — from0037
02C9-02CA	reserved	RUNADR — loader
02CB-02CC	reserved	HIUSED — loader
02CD-02CE	reserved	ZHIUSE — loader
02CF-02D0	reserved	GBYTEA — loader
02D1-02D2	reserved	LOADAD — loader
02D3-02D4	reserved	ZLOADA — loader
02D5-02D6	reserved	DSCTLN — disk sector size
02D7-02D8	reserved	ACMISR — reserved
02D9	reserved	KRPDEL — auto key delay
02DA	reserved	KEYREP — auto key rate
02DB	reserved	NOCLIK — key click disable
02DC	reserved	HELPFG — HELP key flag
02DD	reserved	DMASAV — DMA state save
02DE	reserved	PBPNT — from 001D
02DF	reserved	PBUFSZ — from 001E

02E9	reserved	HNDLOD	—	handler loader flag
02F5	reserved	NEWROW	—	from 0060
02F6-02F7	reserved	NEWCOL	—	from 0061
02F8	reserved	ROWINC	—	from 0079
02F9	reserved	COLINC	—	from 007A
030E	ADDCOR (deleted)	JMPERS	—	option jumpers
0314	TEMP2 moved (0313)	PTIMOT	—	from 001C
033D	reserved	PUPBT1	—	power-up/reset
033E	reserved	PUPBT2	—	power-up/reset
033F	reserved	PUPBT3	—	power-up/reset
03E8	reserved	SUPERF	—	screen editor
03E9	reserved	CKEY	—	from 004A
03EA	reserved	CASSBT	—	from 004B
03EB	reserved	CARTCK	—	cart checksum
03ED-03F8	reserved	ACMVAR	—	reserved
03F9	reserved	MINTLK	—	reserved
03FA	reserved	GINTLK	—	cart interlock
03FB-03FC	reserved	CHLINK	—	handler chain